

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

CROWDSTRIKE, INC.,
Petitioner,

v.

SKYSONG INNOVATIONS, LLC,
Patent Owner.

PTAB Case No. IPR2025-01400
U. S. Patent No. 10,574,721 B2

PATENT OWNER'S PRELIMINARY RESPONSE

TABLE OF CONTENTS

I.	INTRODUCTION	1
II.	RELATED PROCEEDINGS	1
III.	SKYSONG BACKGROUND	1
IV.	THE ASSERTED PRIOR ART	2
A.	<i>Fages</i> (EX1004)	2
B.	<i>Korycki</i> (EX1005)	4
C.	<i>Jai</i> (EX1006)	5
D.	<i>Russello</i> (EX1007).....	5
E.	<i>O’Niell</i> (EX1008)	6
F.	<i>Cardona</i> (EX1009).....	7
G.	<i>Tazuma</i> (EX1010)	8
H.	<i>Hayden</i> (EX1050).....	9
V.	THE ’721 PATENT.....	10
A.	Technical Background.....	10
B.	Prosecution History	11
C.	Priority Date	12
VI.	LEVEL OF ORDINARY SKILL IN THE ART.....	13
VII.	PATENT OWNER’S CLAIM CONSTRUCTIONS	14
A.	“intercept a request” (claim 1).....	14
B.	“browser instance type” (claim 1).....	16

VIII. PETITIONER HAS FAILED TO DEMONSTRATE A REASONABLE LIKELIHOOD OF PREVAILING AS TO ANY CHALLENGED CLAIM.	19
A. GROUND 1: Obviousness in Light of <i>Fages</i> , <i>Korycki</i> , <i>Jai</i> , and <i>Hayden</i>	19
1. Claim 1.....	19
2. Claim 2.....	37
3. Motivation To Combine.....	37
B. GROUND 2: Obviousness in Light of <i>Fages</i> , <i>Korycki</i> , <i>Jai</i> , <i>Hayden</i> and <i>Russello</i>	39
1. Claim 3.....	39
2. Claim 4.....	41
3. Claim 5.....	41
4. Motivation To Combine.....	42
C. GROUND 3: Obviousness in Light of <i>Fages</i> , <i>Korycki</i> , <i>Jai</i> , <i>Hayden</i> , <i>O’Niell</i> , and <i>Cardona</i>	43
1. Claim 6.....	43
2. Claim 7.....	46
3. Claim 8.....	51
4. Claim 12.....	51
5. Claim 13.....	53
6. Claim 16.....	54
7. Claim 17.....	55
8. Claim 18.....	55

9.	Motivation To Combine.....	56
D.	GROUND 4: Obviousness in Light of <i>Fages, Korycki, Jai, Hayden, O’Niell, Cardona, and Russello</i>	57
1.	Claim 9.....	57
2.	Claim 10.....	58
3.	Claim 14.....	58
4.	Claim 15.....	59
A.	GROUND 5: Obviousness in Light of <i>Fages, Korycki, Jai, Hayden, O’Niell, Cardona, and Tazuma</i>	59
1.	Claim 11	60
IX.	CONCLUSION.....	63

TABLE OF AUTHORITIES

CASES

<i>Phillips v. AWH Corp.</i> , 415 F.3d 1303 (Fed. Cir. 2005) (<i>en banc</i>)	15
---	----

TABLE OF ABBREVIATIONS

Patent Owner	Skysong Innovations, LLC (“Skysong”)
Petitioner	CrowdStrike, Inc. (“CrowdStrike”)
PTAB	Patent Trial and Appeals Board
’721 patent	U.S. Patent No. 10,574,721 (EX1001)
<i>Fages</i>	U.S. Patent Application Publication No. 2012/0069401 to Fages et al. (EX1004)
<i>Korycki</i>	U.S. Patent No. 9,009,853 to Korycki et al. (EX1005)
<i>Jai</i>	International Publication No. WO 02/37333 to Jai (EX1006)
<i>Russello</i>	U.S. Patent Application Publication No. 2014/0137184 to Russello et al. (EX1007)
<i>O’Niell</i>	U.S. Patent No. 8,424,017 to O’Niell et al. (EX1008)
<i>Cardona</i>	U.S. Patent Application Publication No. 2003/0061257 to Cardona (EX1009)
<i>Tazuma</i>	U.S. Patent No. 7,493,414 to Tazuma (EX1010)
<i>Hayden</i>	U.S. Patent No. 9,509,783 to Hayden (EX1050)

LIST OF EXHIBITS

Exhibit No.	Descriptions
2001	Skysong's First Amended Complaint against CrowdStrike
2002	Civil Docket For Case # 7:25-cv-00040-DC-DTG (CrowdStrike)
2003	<i>Skysong v. CrowdStrike</i> , Dkt. No. 55, Scheduling Order
2004	Table of CrowdStrike IPRs and Parallel District Litigation Schedules
2005	Civil Docket For Case # 2:25-cv-00098-JRG (Fortinet)
2006	Revision to Rules of Practice Before the Patent Trial and Appeal Board
2007	Executive Order No. 14,028
2008	Declaration of Dr. Nenad Medvidovic

LIST OF CHALLENGED CLAIMS

Limitation	Claim Language
Claim 1	
1	A system, comprising: a processor, operable to:
1.a	receive data defining a first web content class and a second web content class, the first web content class defining a first plurality of Uniform Resource Locators (URLs), and the second web content class defining a second plurality of URLs different than the first plurality of URLs, the second web content class further being associated with sensitive information presenting a security risk;
1.b	generate a first browser instance, the first browser instance defining a browser instance type predetermined to be associated with the first web content class;
1.c	intercept a request to access web content from the first browser instance, the request defining a URL of the second plurality of URLs such that the web content is associated with the second web content class;
1.d	confirm that the URL is not being accessed by another separate already-running browser instance associated with the second web content class, by
1.d.i	implementing a daemon outside the first browser instance that tracks active browser instances and launches a new browser in order to manage web access according to the data defining the first web content class and the second web content class,
1.d.ii	receiving the request via a browser extension associated with the first browser instance,
1.d.iii	forwarding the request from the browser extension to the daemon using a native application,
1.d.iv	confirming, via the daemon that the web content is not associated with the first web content class and that another browser instance is not already launched and available to access the web content; and
1.e	launch a new browser for accessing the web content associated with the second web content class to provide access to the URL of the second plurality of URLs via the new browser.
Claim 2	
2	The system of claim 1, wherein the new browser instance is launched automatically for display, and the web content is associated with a web server.

Claim 3	
3	The system of claim 2, wherein the processor is operable to execute an API call configured to intercept a system call to the web server, the system call to the web server intended to provide access of the URL to the first browser instance.
Claim 4	
4	The system of claim 3, further comprising:
4.a	wherein the processor is operable to pause operations associated with the first browser instance upon interception of the system call; and
4.b	wherein the API controls system calls made by other programs and other browser instances.
Claim 5	
5	The system of claim 1, wherein the processor is operable to implement a tracer to determine whether or not a requested network object is allowed to be accessed by the first browser instance.
Claim 6	
6	The system of claim 1, wherein the daemon comprises a multi-threaded C program that uses Unix domain sockets to communicate with the browser extension and other extensions.
Claim 7	
7	A method for facilitating different categories of content to be displayed in different independent browser instances, the method comprising:
7.a	utilizing a processor in communication with a non-transitory tangible storage medium storing instructions that are executed by the processor to perform operations comprising:
7.a.i	accessing data defining rules for delegating web access associated with a first content class to a first browser type and delegating web access associated with a second content class to a second browser type;
7.a.ii	identifying a request to access an IP address by a first browser instance, the first browser instance being of the first browser type;
7.a.iii	determining that the IP address is associated with the second content class such that the IP address should not be accessed by the first browser instance and confirming that a separate browser instance is not already in use and available to access the request, by:

7.a.iii.1	implementing a daemon, the daemon comprising a multi-threaded program executed outside the first browser instance that tracks active browser instances and launches a new browser instance in order to manage according to the rules,
7.a.iii.2	receiving the request via a browser extension associated with the first browser instance,
7.a.iii.3	forwarding the request from the browser extension to the daemon using domain sockets; and
7.a.iv	generating a second browser instance to access the IP address, the second browser instance being of the second browser type.
Claim 8	
8	The method of claim 7, wherein the second browser instance is launched automatically to a client device for display.
Claim 9	
9	The method of claim 7, further comprising: executing an API call configured to intercept a system call to a web server associated with the IP address, the system call initiated by the first browser instance.
Claim 10	
10	The method of claim 7, further comprising: executing a tracer to determine whether or not a requested network object is allowed to be accessed by the first browser instance.
Claim 11	
11	The method of claim 7, further comprising:
11.a	generating a domain name system (DNS) query to resolve the IP address;
11.b	determining that the IP address is associated with the second content class; and
11.c	issuing a DNS response, the IP address being rewritten in the DNS response to prevent the first browser instance from accessing the IP address and information of the second content class; and
11.d	returning the IP address as rewritten to the first browser instance.
Claim 12	
12	A non-transitory computer readable medium encoded with a set of instructions for facilitating different categories of content to be displayed in different independent browser instances, the set of instructions, executable by a processor, comprising:

12.a	accessing data associating a first web content class with a first browser type and a second web content class with a second browser type;
12.b	accessing a request to access web content via a first browser instance, the first browser instance being of the first browser type; and
12.c	comparing the web content to the first web content class and the second web content class to determine whether the web content is accessible to the first browser instance, by
12.c.i	implementing a daemon, the daemon comprising a multi-threaded program executed outside the first browser instance that tracks active browser instances and launches a new browser instance in order to manage according to the rules,
12.c.ii	receiving the request via a browser extension associated with the first browser instance,
12.c.iii	forwarding the request from the browser extension to the daemon using domain sockets.
Claim 13	
13	The non-transitory computer readable medium of claim 12, further comprising:
13.a	generating a second browser instance to access the web content,
13.b	wherein the second browser instance is launched automatically to a client device for display.
Claim 14	
14	The non-transitory computer readable medium of claim 12, further comprising:
14.a	executing an API call configured to intercept a system call to a web server,
14.b	the system call associated with the first browser instance.
Claim 15	
15	The non-transitory computer readable medium of claim 12, further comprising: executing a tracer to determine whether or not the first browser instance is authorized to access a requested network.
Claim 16	
16	The non-transitory computer readable medium of claim 12, wherein the second web content class is associated with sensitive information indicating that accessing the web content by the first browser instance presents a security risk.

Claim 17	
17	The non-transitory computer readable medium of claim 12, wherein the first web content class is associated with a first plurality of URLs and the second web content class is associated with a second plurality of URLs, the first plurality of URLs being different from the second plurality of URLs.
Claim 18	
18	The non-transitory computer readable medium of claim 12, wherein the data further defines rules limiting web access by the first browser instance to URLs defined by the first web content class.

I. INTRODUCTION

Patent Owner Skysong respectfully submits this Preliminary Response to the Petition for *Inter Partes* Review dated September 12, 2025 (“Petition”) of U.S. Patent No. 10,74,721 (EX1001). This response is timely filed. *See* Paper 3.

II. RELATED PROCEEDINGS

The ’721 patent is the subject of the following related litigation:

- *Skysong Innovations, LLC v. CrowdStrike, Inc. and CrowdStrike Holdings, Inc.*, Case No. 7:25-cv-00040, filed January 31, 2025, in the United States District Court for the Western District of Texas; and
- *Skysong Innovations, LLC v. Fortinet, Inc.*, Case No. 2:25-cv-00098, filed January 31, 2025, in the United States District Court for the Eastern District of Texas.

III. SKYSONG BACKGROUND

Skysong is a nonprofit organization, and the exclusive intellectual property management and technology transfer organization for Arizona State University (“ASU”). Skysong works closely with ASU and its technology industry partners to translate ASU’s innovations into commercial applications, fostering economic development and public benefit. *See* EX2001 (Skysong’s First Amended Complaint), ¶¶3–4. Skysong significantly contributes to the mission statement of

the University and has become a core component of Arizona's innovation ecosystem and a driver of high-tech industry in the state.

The '721 patent stems from research conducted by Dr. Rida A. Bazzi and his team. It discloses systems and methods for securing access to sensitive content on the web which automates transparent compartmentalization practices for accessing different kinds of content with different browser instances. *See* EX1001, abstract.

IV. THE ASSERTED PRIOR ART

Dr. Nenad Medvidovic, Department Chair of the Computer Science Department of the University of Southern California and leading cybersecurity authority, has been retained by Skysong as an independent expert to assist in the technical analysis of the '721 patent and the asserted prior art. He has provided a declaration (EX2008, "*Declaration Of Nenad Medvidovic, Ph. D., In Support Of Patent Owner's Preliminary Response*") setting forth his opinions, including a synopsis of the relevant aspects of the asserted prior art. His synopsis of the asserted prior art is relevant and reproduced here for reference:

A. *Fages* (EX1004)¹

Fages refers to U.S. Patent Application Publication No. 2012/0069401 to Fages et al.

¹ EX2008 at ¶¶95–99 (*Fages*).

Fages discloses a server-based remote browsing method that isolates web activity into independent execution environments on an access server. The user operates a remote browser via a client module, while only screen images are sent to the user station, so scripts execute solely on the server. This design mitigates first-type of attacks, targeting local devices. The first type of attack seeks to attack the user station used to access the malicious website. *Fages* at [0012].

Fages uses a pre-recorded list of sensitive URLs for which access from the current environment is blocked. Instead, control switches to a second browser instance running in a different execution environment.

Importantly, *Fages* discloses a server (20) capable of instantiating multiple instances of a web browser, and coordinating use thereof by the user's remote stations. *Fages* at [0063]. The server "is capable of simultaneously executing several Internet browsers in distinct execution environments." *Id.* "The server 20 also has a server module 40 of a software program for the remote control of the Internet browsers executed by the server 20." *Id.* at [0064]. "The server 20 is connected to a memory 44 containing the instructions and information needed to execute the method of FIG. 6." *Id.* at [0065]. In particular, the memory 44 has a buffer memory 46 and files 48 for configuring different possible Internet sessions." *Id.* Thus, the server itself is capable of executing and tracking several Internet browsers, and their specific use. Therefore, *Fages* has no need for a system daemon

to keep track of the multiple browser instances. Similarly, *Fages*' server is capable of communicating directly with any of the browser instances. This also obviates the need for a "browser extension" and "native application" to facilitate communications between the browser instance and the system daemon as claimed in the '721 patent.

B. *Korycki* (EX1005)²

Korycki refers to U.S. Patent No. 9,009,853 to Korycki et al. *Korycki* was a reference cited in the '721 patent.

Korycki discloses a client-side framework that enables secure communication between web applications running on the same device but in separate browser instances.

The daemon checks an Access Control List (ACL) of approved partner domains maintained by the service provider from a database (queried from a provider server and cached locally). If the domain is approved, the daemon permits the channel subscription/publication and routes messages to subscribed plugins; otherwise, it blocks the interaction. This architecture enforces policy locally, reducing latency, server load, and enabling offline operation via cached ACLs.

The essence of the disclosure of *Korycki* is a client-side framework not a server, thus it teaches away from combining its disclosure with a server-based disclosure such as *Fages*.

² EX2008 at ¶¶100–103 (*Korycki*).

C. *Jai* (EX1006)³

Jai refers to International Publication No. WO 02/37333 to Jai.

Jai discloses a system and method that help users remember and efficiently use browser bookmarks by providing contextual information. When a user enters a URL or visits a webpage, the browser checks if the address or any hyperlinks on the page match existing bookmarks. If matches are found, the browser displays visual or audio indicators to alert the user. The system also builds a context network by recording other webpages that contain hyperlinks to bookmarked sites, creating a repository of related URLs. This enables the browser to display lists of related sites or perform automatic searches to find additional information about a bookmarked topic. Search results and linked pages can be shown in drop-down menus for easy access. The invention also prevents multiple browser windows from opening the same site, improving efficiency. It can be applied to desktop or mobile browsers for enhanced bookmark management and browsing context.

D. *Russello* (EX1007)⁴

Russello refers to U.S. Patent Application Publication No. 2014/0137184 to Russello et al.

³ EX2008 at ¶¶104–105 (*Jai*).

⁴ EX2008 at ¶¶106–107 (*Russello*).

Russello discloses a method and system for enforcing application-specific security policies on Linux-based devices. The invention monitors applications at the process level, starting from process creation, and attaches a security monitor to each process to intercept system calls and library calls. Each monitor verifies that a process's behavior conforms to its associated security policy before allowing the kernel to execute a call. The system can employ two layers of monitoring: Function Call Interposition (FCI) for user-level function calls and System Call Interposition (SCI) for kernel-level system calls, depending on the sensitivity of the process. Security policies define permitted operations for each application and can be stored locally or updated remotely. This approach enables fine-grained runtime enforcement without modifying the kernel, offering enhanced protection against malicious or unauthorized application behavior on Linux-based devices such as smartphones, tablets, and embedded systems.

E. *O'Niell* (EX1008)⁵

O'Niell refers to U.S. Patent No. 8,424,017 to O'Niell et al.

O'Niell discloses a method and system for determining whether a process in an information handling system (IHS) is active (current) or inactive (stale). When an application launches a process, the operating system creates a domain socket such as a Unix or Linux socket and activates a process ID embedding handler. This

⁵ EX2008 at ¶¶108–109 (*O'Niell*).

handler retrieves the process's name and ID, then embeds both into the socket's pathname (e.g., "ProcessName_PID.pid"). The embedded pathname is bound to the socket and recorded in a status file (e.g., /proc/net/unix). To verify if a process is still running, a socket pathname checker compares the embedded pathname against entries in the status file. If a match is found, the process is active; if not, it is stale or terminated. This approach enables fast, accurate detection of process status without exhaustive directory searches, improving efficiency in process management within operating systems and multi-process computing environments.

F. *Cardona* (EX1009)⁶

Cardona refers to U.S. Patent Application Publication No. 2003/0061257 to Cardona.

Cardona describes a multithreaded universal daemon (MUD) for networked devices that consolidates protocol-independent socket management and task concurrency into a reusable library, minimizing resource use, especially on embedded devices. A developer invokes a single create function (and optional edit function) with protocol-specific parameters (port, handlers, thread count, memory pools, timeouts, etc.) to generate a daemon that spawns multiple identical tasks/threads. Tasks use a lead-token scheme so only one task blocks on socket I/O at a time; when data arrives a task dispatches an application-specific connection

⁶ EX2008 at ¶¶110–111 (*Cardona*).

handler (initialization, data, or error handler) and then passes the token to the next task. The daemon maintains socket states (IDLE, BUSY, CLOSED) and supports many concurrent connections without binding threads to sockets. By centralizing common daemon functions and exposing handler hooks, the MUD reduces code duplication, conserves memory, and simplifies creating protocol-specific servers (e.g., Telnet, HTTP, Java applet) across constrained devices.

G. *Tazuma* (EX1010) ⁷

Tazuma refers to U.S. Patent No. 7,493,414 to Tazuma.

Tazuma discloses a system and method for web browser redirection that ensures reliable user access even when normal DNS resolution or network connectivity fails. Conventional redirection systems stop working if a browser requests a non-resolvable domain name or if the upstream network is unavailable. The invention solves this by assigning a fake IP address to the browser whenever a domain cannot be resolved. When the browser sends a request to this fake IP, the system detects it and redirects the browser to a specific page, such as a local portal, login page, or user-friendly error page. The system employs components like a gateway device, packet filter, and web request handler to manage both direct and proxy-mode browser configurations. It also handles mapped network drives and supports redirection during pre-login and post-login phases. This method is

⁷ EX2008 at ¶¶112–113 (*Tazuma*).

particularly useful in mobile or intermittently connected networks, ensuring seamless and intelligent browser redirection.

H. *Hayden* (EX1050)⁸

Hayden refers to U.S. Patent No. 9,509,783 to Hayden.

Hayden discloses a network-computing provider to create and deploy customized browser virtual machine images that are pre-optimized for particular websites or types of web content. For example, when a user requests a site that is heavy on video, the system instantiates a browser VM that is fully configured (pre-cached resources, compiled scripts, tuned settings) to render that site rapidly. The architecture includes a virtual machine manager with modules for image creation, forecasting/deployment, instance selection, and analysis/optimization of traffic patterns. When demand for a specific site rises, the provider dynamically generates or deploys additional specialized browser VMs to serve it. Client requests are routed to the best-fitting image via an instance selector. By customizing browsers in this way (rather than using a generic one), the system reduces latency, improves performance, and optimizes resource usage in large-scale cloud browsing environments.

⁸ EX2008 at ¶¶114–115 (*Hayden*).

V. THE '721 PATENT

A. Technical Background

Dr. Medvidovic provides a helpful explanation of the technical background of the '721 patent in paragraphs 71–74 of his declaration, reproduced in full here:

71. The '721 patent, titled “*Systems And Methods For An Automatic Fresh Browser Instance For Accessing Internet Content*,” claims priority to provisional application No. 61/912,716, filed on December 6, 2013. '721 patent at 1:8–10. The patent addresses securing access to sensitive content on the web. *Id.* at abstract. Its approach automates transparent compartmentalization practices for accessing different kinds of content with different browser instances. *Id.*

72. The '721 patent teaches “an automatic fresh browser instance (‘FBI’) system.” *Id.* at 2:29–30. In one embodiment “the FBI system enables the processing of two types of sites for a novice user: sensitive sites and every other type of site. Access to a sensitive site is done through a dedicated FBI. Access to all other sites is done through separate browser instance(s) as it would be done without Auto-FBI. When a user attempts to access a sensitive website by clicking on a link or entering a URL directly in the address bar, an FBI is created to handle the access. When the user finishes accessing the sensitive site, the user can close the FBI or minimize it. If the user later attempts to access the sensitive site from another browser instance, a new instance is activated again, or, if there is an existing active instance for the site, that instance is used to process the request.” *Id.* at 4:35–48. “In order to keep track of all active browser instances and their conflict of interest (‘COI’) classes, a daemon outside the browser keeps track of active instances and launches new instances as needed.” *Id.* at 2:43–46.

73. The '721 patent addressed the shortcomings of the prior art by “automat[ing] compartmentalization practices for accessing different kinds of content with different browser instances. The automation is transparent to the user and does not require any modification of how non-sensitive content is accessed.” *Id.* at abstract.

74. The system has two main components. “One component, an enforcer, intercepts system calls and modifies them as needed to ensure that

sensitive content is only accessed through the appropriate instance of the browser.” *Id.* at 7:8–11. “Another component, an enabler determines when sensitive content is requested and, if needed, launches a new fresh instance of the browser to access the content.” *Id.* at 7:11–14.

EX2008 at ¶¶71–74.

B. Prosecution History

The provisional application that resulted in the ’721 patent was filed on December 6, 2013, and the corresponding non-provisional application on December 8, 2014. EX1001.

On December 23, 2016, a non-final office action issued rejecting all claims. EX1002 at 191.

On March 22, 2017, an amendment was submitted in response to the preliminary rejection. *Id.* at 213. The claims were amended to overcome *Shephard* (US 9,055,048). *Id.* at 214. Applicant specifically differentiated its invention over *Shephard* based on intercepting system calls. *Id.* at 220.

On May 19, 2017, a final rejection issued, rejecting all claims. *Id.* at 225.

On August 21, 2017, another response was submitted with argument relating to the second content class that includes sensitive information presenting a security risk, confirming that the URL is not associated with the first content class, and confirming existence of another already-running browser instance associated with the second class. *Id.* at 246.

On September 6, 2017, an examiner-initiated interview was held. *Id.* at 261.

On November 13, 2017, a request for continued examination was filed and amendment was submitted. *Id.* at 272.

On March 8, 2018, another non-final rejection issued. *Id.* at 284.

On August 8, 2018, the claims were further amended. *Id.* at 305; *see also id.* at 321 (corrected amendment filed August 14, 2018). Applicant noted that the *Loesch* reference (US 8,095,967) does not teach “confirm[ing] that the URL is not being accessed by an already-running browser instance associated with the second web content class.” *Id.*

On December 18, 2018, another non-final rejection issued, rejecting all claims. *Id.* at 332.

On April 8, 2019, an applicant initiated interview was held. *Id.* at 354.

On May 20, 2019, the claims were further amended, including implementing a daemon. *Id.* at 362.

On September 5, 2019, a notice of allowance issued. *Id.* at 373. The examiner specifically noted that the prior art did not teach or suggest the amended claims. *Id.* at 378 (citing amendments and remarks filed May 20, 2019).

C. Priority Date

The '721 patent derives priority from provisional application No. 61/912,716, filed on December 6, 2013. The provisional application fully discloses the subject

matter of the claims in the '721 patent. EX2008 at ¶76. Petitioner alleges the following priority dates for the asserted prior art:

- a. *Fages* (EX1004): March 22, 2012 (Pet. at 5)
- b. *Korycki* (EX1005): March 7, 2012 (Pet. at 5)
- c. *Jai* (EX1006): May 10, 2002 (Pet. at 6)
- d. *Russello* (EX1007): October 10, 2013 (Pet. at 7)
- e. *O'Niell* (EX1008): November 3, 2011 (Pet. at 8)
- f. *Cardona* (EX1009): March 27, 2003 (Pet. at 8)
- g. *Tazuma* (EX1010): February 17, 2009 (Pet. at 9)
- h. *Hayden* (EX1050): January 26, 2012 (Pet. at 7)

Since these dates predate the provisional patent application as well, Patent Owner's analysis relies on the specification of the '721 patent itself. However, Patent Owner is still investigating the veracity of Petitioner's alleged dates. To the extent these dates are inaccurate and in fact post-date the provisional application, Patent Owner reserves the right to identify the respective portions of the provisional application supporting the claims of the '721 patent.

VI. LEVEL OF ORDINARY SKILL IN THE ART

In consideration of the type of problems encountered in the art, the prior solutions to those problems found in the prior art references, the rapidity with which innovations are made, the sophistication of the technology, the level of education of

active workers in the field, and in reliance on Dr. Medvidovic’s expert opinion, Patent Owner defines the relevant skill in the art as follows: A person of ordinary skill in the art (“POSITA”) at the time of the filing of the ’721 patent would have had at least a Bachelor’s degree in computer science or computer engineering, and 2–4 years of experience working in the field of software development, distributed systems, data mining, and cybersecurity. A graduate degree in computer science, computer engineering, artificial intelligence, cybersecurity, or software engineering may be substituted wholly or in part for work experience. EX2008 at ¶38. Dr. Medvidovic qualifies as a POSITA. *Id.* at ¶39.

VII. PATENT OWNER’S CLAIM CONSTRUCTIONS

Patent claims are construed using the framework set forth in *Phillips v. AWH Corp.*, 415 F.3d 1303, 1312–15 (Fed. Cir. 2005) (*en banc*). CrowdStrike purports to rely on the plain and ordinary meaning of all claim terms. Pet. at 4–5. However, CrowdStrike misinterprets two key claim terms. Patent Owner sets forth these two terms and their proper construction below. For all remaining terms, Patent Owner applies the ordinary meaning of the terms as a POSITA would have understood them at the time of the invention.

A. “intercept a request” (claim 1)

The term “intercept a request” is properly construed as “intercept a system call.” EX2008 at ¶117. In context, claim 1 includes the limitation “intercept a

request to access web content from the first browser instance.” ’721 patent, claim

1. As Dr. Medvidovic explains, “the ’721 patent identifies three different aspects of its enforcement mechanism: (1) enforcement inside the browser, (2) enforcement in a browser extension, and (3) enforcement outside the browser.” EX2008 at ¶118 (*citing* ’721 patent at 6:55–7:6). The first two approaches are internal to the browser, and therefore have no need to “intercept a request” to access web content. “Only the third approach—enforcement outside the browser—requires intercepting a request because it is external to the browser.” EX2008 at ¶118.

The ’721 patent explicitly discloses an enforcer and an enabler. ’721 patent at 7:8. The enforcer “intercepts system calls and modifies them as needed to ensure that sensitive content is only accessed through the appropriate instance of the browser.” *Id.* at 7:8–11. The intercepting aspect of the enforcer is described in the section ‘Intercepting System Calls.’ *Id.* at 7:16–44. Specifically, the patent teaches that “to intercept system calls, the system includes a ptrace API [Application Programming Interface] available in Linux.” *Id.* at 7:17–18. “This API allows one program (tracer) to be notified and take control of every system call made by another program (browser).” *Id.* at 7:18–20. “Using this API enables us to control every interaction between the browser and the operating system.” *Id.* at 7:28–29.

One of the Petitioner’s references, *Russello*, confirms that to “intercept” means “to intercept all or at least a set of system calls executed by the target process

and report those to the master process.” *Russello* at [0026]; *see also, e.g.*, [0039] (“intercept all or at least a set of system calls executed by the mother process and report those to the main monitor”); [0091] (“intercept system calls made to the kernel by the code and direct those to a monitoring entity to implement a security action”); [0092] (“any system calls invoked by the new child process are intercepted and directed to the embedded process monitor of the parent process”); and [0114] (“intercepting the system call invocations to the kernel”).

Therefore, the plain and ordinary meaning of the term “*intercept a request*” [to access web content from the first browser instance] as understood by a POSITA at the relevant time is to “intercept a system call” issued by the browser.

B. “browser instance type” (claim 1)

The term “browser instance type” is properly construed as “the type of internet browser used for the instance.” EX2008 at ¶122. The ’721 patent provides Google Chrome and Mozilla Firefox as examples of internet browser types. ’721 patent at 5:65. These exemplary browser instance types are used in the example of figure 3:

To allow for using different browsers for different classes, e.g., using Google Chrome for e-commerce and Mozilla Firefox for banking, we specify the browser type rule that should be followed for each class with a type descriptor (that comes immediately after a in the sample file):

‘F’: Mozilla Firefox instance should be used

‘C’: Google Chrome instance should be used

‘=’: the launched browser instance should be the same type as the launcher

‘!’: the launched browser instance type should be different from that of the launcher

‘_’: places no restriction on the type of the browser

The implicit general class has the implicit type descriptor ‘_’, so both Google Chrome and Mozilla Firefox can be used to browse websites from the general class. While Google Chrome and Firefox have been provided as examples, it is contemplated that any type of internet browser, or any browser accessible by a user, may be used.

Id. at 5:51–6:2. The browser type rule allows the user to define the preferred type of browser for each COI class.

Dr. Medvidovic provides an example in which the ’721 patent describes the use of these browser instance types in reference to figure 3, which shows three exemplary conflict of interest (“COI”) class definitions:

The three COI classes have different type descriptors, which determine which browser type will be used for that class. “The example file shown in Fig. 3 has two explicit COI classes [C₁ and C₂]. The first class C₁ **302** has type descriptor ‘!’ and the second class C₂ **304** has type descriptor ‘C.” *Id.* at 6:3–6. This type descriptor determines which browser instance type is used for that particular class. In the example, the user is browsing using Google Chrome. *Id.* at 6:8–9. First, “an instance of Google Chrome of the general class is created.” *Id.* “The user can navigate to any site that is not listed in [C₁ and C₂] in that browser instance.” *Id.* at 6:10–11. However, for any URL listed in class C₁ (such as www.bankofamerica.com), “a fresh instance of Mozilla Firefox will be created.” *Id.* at 6:14. “That is because the type descriptor for [C₁] specifies that an instance with a different type from that of the launcher (Google Chrome in this example) should be used.” *Id.* at 6:14–17. For any URL listed in class C₂ (such as www.ebay.com) on the other hand, “a new instance of Google Chrome will be launched to handle the request regardless of the browser type used to initiate the request.” *Id.* at 6:21–25. Thus, browser instance type refers to the type of browser that is permitted or desired for the desired URL.

EX2008 at ¶123 (citing '721 patent at 6:3–25 and Fig. 3). Thus, the user is able to predefine which browser instance type the system should use to open the desired website.

Petitioner confuses browser type with website type, and erroneously interprets browser type as “(1) a secure browser instance/browser instance type and (2) an unsecured browser instance/browser instance type” used for sensitive content and general content, respectively. Pet. at 19. In support, Petitioner cites the abstract and the general description of the system for a novice user at 4:33–5:21. Neither support Petitioner’s interpretation. First, the abstract does not reference a “type” at all, and merely states that different browser instances are used for different URLs. Second, the only reference to “type” in the cited novice user section relates to web site type, not browser type:

[T]he FBI system enables the processing of *two types of sites* for a novice user: sensitive sites and *every other type of site*. Access to a sensitive site is done through a dedicated FBI. Access to all other sites is done through separate browser instance(s) as it would be done without Auto-FBI.

'721 patent. at 4:34–36 (emphasis added). The emphasized portions are the only reference to “type” in that section. Although these different types of web sites can be shown in different browser instances, this disclosure says nothing about type of browser used in any given instance. The only reference to a “browser instance type” in the patent relates to the ability to use different browsers for different classes, e.g.,

using Google Chrome for e-commerce and Mozilla Firefox for banking, as explained above.

Petitioner further points to the prosecution history to support its interpretation, and argues that the examiner [mis]interpreted the claim element. Pet. at 19. This is not supported by the file history. First, the cited examiner statements rejecting the claim do not address the browser instant type. The examiner simply referred to the use of general and private browser instances. In any event, even if the examiner applied a faulty interpretation, nothing in the file history supports that the applicant agreed therewith.

Therefore, the plain and ordinary meaning of the term “browser instance type” as understood by a POSITA at the relevant time is “the type of internet browser used for the instance.”

VIII. PETITIONER HAS FAILED TO DEMONSTRATE A REASONABLE LIKELIHOOD OF PREVAILING AS TO ANY CHALLENGED CLAIM.

A. GROUND 1: Obviousness in Light of *Fages*, *Korycki*, *Jai*, and *Hayden*

Petitioner asserts that claims 1 and 2 are obvious in light of *Fages*, *Korycki*, *Jai*, and *Hayden*.

1. Claim 1

Claim 1. A system, comprising: a processor, operable to:

a) Claim 1.b: generate a first browser instance, the first browser instance defining a browser instance type predetermined to be associated with the first web content class;

Petitioner interprets the “browser instance type” as browser instances for secure versus non-secure data, relying in part on the ’721 patent’s “novice user” embodiment where one instance handles “sensitive” and another “general” content. Pet. at 18–19. This interpretation of “browser instance type” is incorrect.

As detailed above in the claim construction section for “browser instance type,” a POSITA would understand the term “browser instance type” as “the type of internet browser used for the instance,” such as the listed examples of Google Chrome and Mozilla Firefox. EX2008 at ¶131 (citing ’721 patent at 5:65). Under the proper construction, this functionality is not taught or suggested by *Fages*.

As Dr. Medvidovic explains, “the ’721 patent clearly defines ‘internet browser type’ ‘to allow for using different browsers for different classes, e.g., using Google Chrome for e-commerce and Mozilla Firefox for banking.’” EX2008 at ¶131. For that reason, the ’721 patent specifies “a browser type rule that should be followed for each class with a type descriptor:”

‘F’: Mozilla Firefox instance should be used

‘C’: Google Chrome instance should be used

‘=’: the launched browser instance should be the same type as the launcher

‘!’: the launched browser instance type should be different from that of the launcher

‘_’: places no restriction on the type of the browser

’721 patent at 5:51–6:2. The patent further teaches that “while Google Chrome and Firefox have been provided as examples, . . . any type of internet browser, or any browser accessible by a user, may be used.” *Id.*

Dr. Medvidovic identifies a specific example of this feature provided in the patent, in figure 3:

The figure shows three COI classes with different “type” descriptors, which determine which browser type will be used for that class. In the example, the user is browsing using Google Chrome. *Id.* at 6:8–9. “The user can navigate to any site that is not listed in [C₁ and C₂] in that browser instance.” *Id.* at 6:10–11. However, for any URL listed in class C₁ (e.g., www.bankofamerica.com), “a fresh instance of Mozilla Firefox will be created.” *Id.* at 6:14. For any URL listed in class C₂ (e.g., www.ebay.com) on the other hand, “a new instance of Google Chrome will be launched to handle the request regardless of the browser type used to initiate the request.” *Id.* at 6:21–25. Thus, browser instance type refers to the type of browser that is permitted or desired for the desired URL.

EX2008 at 132 (citing ’721 patent at 6:3–25 and Fig. 3).

Petitioner confuses “browser type” with “website type.” Pet. at 19. In support, Petitioner cites the patent abstract, and the general description of the system for a novice user at 4:33–5:21. However, neither support Petitioner’s construction. The abstract does not reference a “type” at all, and the only reference to “type” in the cited novice user section relates to web site type, not browser type:

[T]he FBI system enables the processing of two types of sites for a novice user: sensitive sites and every other type of site. Access to a sensitive site is done through a dedicated FBI. Access to all other sites

is done through separate browser instance(s) as it would be done without Auto-FBI.

'721 patent at 4:34–36 (emphasis added). The emphasized portions are the only reference to “type” in that section. This disclosure says nothing about the type of browser used in any given instance. The only reference to a “browser instance type” in the patent relates to the invention’s ability to allow different browsers for different classes, e.g., using Google Chrome for e-commerce and Mozilla Firefox for banking, for advanced users, as explained above.

Petitioner’s reference to the prosecution history also does not support its interpretation. Pet. at 19. First, the examiner did not address the *browser instant type*, but instead simply referred to the use of general and private browser instances. Second, even if the examiner applied a faulty interpretation, nothing in the file history supports that the applicant agreed therewith. Instead, the applicant specifically described managing content among “a plurality of different browser types.” EX1002 at 370. Thus, the prosecution history supports patent owner’s interpretation.

“In the ’721 patent, “browser instance type” is explicitly tied to a web content COI class, consistent with the experienced-user embodiment.” EX2008 at ¶133. *Fages* “only discloses using different browser environments for secure vs. non-secure sessions.” *Id.* It does not teach or suggest a browser instance type predetermined to be associated with a web content class as claimed.

b) Claim 1.c: intercept a request to access web content from the first browser instance, the request defining a URL of the second plurality of URLs such that the web content is associated with the second web content class;

Petitioner claims that “*Fages*’ server module 40 ‘*intercept[s]*’ requests to access sensitive/secure web content that are sent to server 20 from client module 52.” Pet. at 24 (citing *Fages*, [0072]–[0074], [0086]–[0088], [0090]–[0093], [0098]–[0099], [0109], and *Dec.*, ¶¶185–187). However, as Dr. Medvidovic notes, this interpretation of the phrase “intercept a request” is directly contradicted by Petitioner’s own interpretation of the term “intercept” regarding claim 3:

The ’721 Patent describes intercepting system calls using a ptrace API. ’721 Patent, 7:16-31 (“This API allows one program (tracer) to be notified and take control of every system call made by another program (browser).”), 8:62-65, 9:22–26. The ’721 Patent claims the API intercepts system calls (claim 3) and controls system calls made by other programs and other browser instances (claim 4). ’721 Patent, Claims 3-4.

EX2008 at ¶134 (citing Pet. at 53). In reference to claim 3, Petitioner relies on the interpretation the ’721 patent uses for that phrase. In contrast, regarding claim 1, Petitioner points only to “requests to access sensitive/secure web content that are sent to server 20 from client module 52.” Petitioner identifies nothing in *Fages* that teaches or suggests intercepting a system call.

Further, the cited portion of Dr. Zadock’s declaration does no more than cite the same portions of *Fages* as in the Petition without explanation. EX1003 at ¶¶185–186 (no reference to “intercept”), and ¶187 (citing *Fages* without explanation).

Second, of the cited portion of *Fages*, only [0086]-[0088] relate to this operation, as detailed below. The remaining citations relate to other facets of *Fages*. See *Fages* at [0072]-[0074] (discusses Figs. 2, 3, and 4, regarding creation of lists of URLs), [0090]-[0093] (discusses operation of the system *after* the new browser instance is established), [0098]-[0099] (discloses comparing the new session request to the URL lists), and [0109] (discloses the option of either blocking the new request or opening a new browser instance in another execution environment). None of these relate to intercepting or even transmitting the request.

Fages does not disclose “intercept[ing] a request” as claimed. EX2008 at ¶135. Dr. Medvidovic explains that *Fages* uses a different mechanism, in reference to its figure 6:

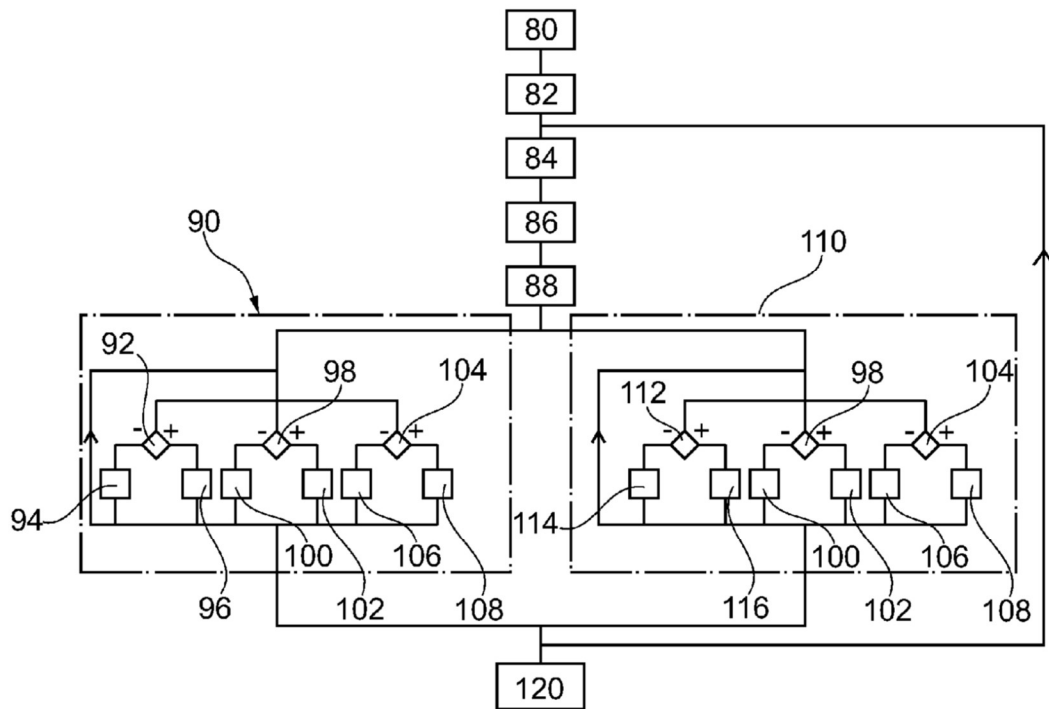


Fig. 6

Id. (citing *Fages*, Fig. 6). “First, in step 80, an administrator configures server 20.”

Id. (citing *Fages* at [0085]). *Fages* then details the process:

[0086] . . . At a step 82, when a user wishes to connect to the Internet 4, he launches the execution of the module 52 on his user station. The module 52 then links up to the module 40 and automatically downloads the list of Internet sessions defined in the table 70. This list is presented to the user by means of the man/machine interface 54.

[0086] At a step 84, the module 52 acquires the identifier S, of the session selected by the user of the station 26 and transmits this identifier to the module 40.

[0087] In response, at a step 86, the server 20 creates an execution environment and launches the execution in this environment of the Internet browser identified and configured as indicated in the list C, associated with the identifier S, acquired during the step 84.

Id. (citing *Fages* at [0086]–[0088]). “These steps accomplish *Fages*’ goal: ‘[t]hus, a sensitive site such as an Internet site enabling the management of a user’s bank accounts will be accessible only by means of an Internet browser whose configuration defined in the list C, is appropriate to this use.’” *Id.* (citing *Fages* at [0088]).

Dr. Medvidovic further provides an explanation for why *Fages*’ process does not intercept “a request to access web content” or a system call as claimed: “The reason is simple. *Fages* offers a secure specialized server implementation for offering browsing security. It is not aimed at transparent compartmentalization practices on a user’s machine.” EX2008 at ¶136. Therefore, *Fages* was able to design its system to allow messages between a remote user terminal and server to accomplish its goal. The ’721 patent, on the other hand, is aimed to transparently operate on the user’s station itself. “Transparently” means that the user’s experience should not change, which is the reason the invention teaches intercepting system calls. Using that technique, no modifications to the user’s experience are necessary.

Dr. Medvidovic confirms this analysis:

Unlike *Fages*, the ’721 patent compartmentalization “automation is transparent to the user and does not require any modification of how non-sensitive content is accessed.” ’721 patent abstract. It is this transparent operation aspect of the ’721 patent that requires that system calls are “intercepted.” This allows the user software to operate as normal, without modification. *Fages* has no such requirement because it has full control of the entire system, server and user side.

EX2008 at ¶137.

Notably, the “intercept” requirement was explicitly used to differentiate prior art during prosecution of the ’721 patent. EX1002 at 220.

Petitioner identifies nothing in the prior art that teaches or suggests this claim element as properly understood by a POSITA at the relevant time.

c) Claim 1.d: confirm that the URL is not being accessed by another separate already-running browser instance associated with the second web content class, by

Petitioner admits that “*Fages* does not expressly discuss confirming that the URL is not being accessed by another separate already running browser instance associated with the second web content class.” Pet. at 25. Petitioner then identifies *Jai* as teaching “‘compar[ing] any new URL to be accessed to any other URLs’ that are currently open/active ‘to determine if such new URL is already being accessed by other browsers,’” in order to avoid “duplicate applications of the same site.” *Id.* at 25 (citing *Jai* at abstract and 3:22–4:2). *Jai* does not teach or suggest this claim element.

“Unlike the ’721 patent, *Jai* only checks whether a given ‘new URL’ is already open.” EX2008 at ¶140. “*Jai* explicitly only “prevents redundant applications of the same website 20 from being opened.” *Id.* (citing *Jai* at 12:8–14).

As Dr. Medvidovic explains, the ’721 patent implements a differing solution. “The ’721 patent does not check whether the new URL is already being accessed,

but instead explicitly checks whether any URL in the same COI class is already being accessed by another separate already-running browser instance.” *Id.* at ¶145. This requirement is explicitly claimed. “The reason for this is that the ’721 patent does not aim to prevent the same URL being opened multiple times, as one might do to review different portions of a reference website in separate windows.” *Id.* “Instead, the ’721 patent performs this check because all URLs in a given COI class can be accessed in the same browser instance.” *Id.* (citing ’721 patent at 2:64–67 (allowing multiple sites in the same COI instance) and 3:8–11 (“allowing an experienced user to specify COI classes of content that can coexist in a browser instance”)). “This is the whole purpose of COI classes, and saves system resources.” *Id.* (citing ’721 patent at 5:23–33). For example, all secured banking websites are in the same COI class and can therefore be accessed in the same browser instance, as shown in the example of figure 3 and related description. *Id.* at Fig. 3 and 6:3–25.

Similarly, *Hayden* only determines “if a particular browser instance is already launched” “corresponding to the requested web site.” Pet. at 28, 29 (citing *Hayden* at 35:38-62). As Dr. Medvidovic explains, “*Hayden* only seeks to identify a browser instance that is ‘customized for the requested web site.’” (EX2008 at ¶142 (citing Pet. at 29 (citing *Hayden* at 36:7–11))). Nothing therein discloses checking whether “the URL is not being accessed by another separate already-running browser

instance associated with the second web content class” as claimed. ’721 patent, claim 1.

Notably, this claim limitation was also used to differentiate prior art during prosecution of the ’721 patent. *See, e.g.*, EX1002 at 260, 312, and 329.

Petitioner identifies nothing in the prior art that teaches or suggests this claim element as properly understood by a POSITA at the relevant time.

d) Claim 1.d.i: implementing a daemon outside the first browser instance that tracks active browser instances and launches a new browser in order to manage web access according to the data defining the first web content class and the second web content class,

Petitioner admits that “*Fages-Jai-Hayden* does not expressly discuss utilizing a daemon.” Pet. at 36. Petitioner thus proposes adding *Korycki* to the combination. However, as Dr. Medvidovic explains, “the functionality provided by *Korycki* is not necessary for the *Fages-Jai-Hayden* combination because *Fages* is able to monitor its browser instances directly, and therefore, has no need to utilize a background daemon process for monitoring.” EX2008 at ¶144. For that reason, “a POSITA would not be motivated to incur the additional overhead and complexity of implementing a daemon.” *Id.*

To illustrate the technical difference, Dr. Medvidovic explains the operation of *Fages*:

Fages’ server (20) directly instantiates multiple instances of a web browser, and coordinates use thereof. *Fages* at [0063]. The server “is

capable of simultaneously executing several Internet browsers in distinct execution environments.” *Id.* “The server 20 also has a server module 40 of a software program for the remote control of the Internet browsers executed by the server 20.” *Id.* at [0064]. “The server 20 is connected to a memory 44 containing the instructions and information needed to execute the method of FIG. 6.” *Id.* at [0065]. In particular, “the memory 44 has a buffer memory 46 and files 48 for configuring different possible Internet sessions.” *Id.* Thus, the server itself is capable of executing and tracking several Internet browsers and their specific use.

EX2008 at ¶145. Since *Fages*’ user stations are able to directly message the server, “*Fages* has no need for a system daemon to keep track of the multiple browser instances.” *Id.* Similarly, “because *Fages*’ server is capable of communicating directly with the browser instances, it does not require a daemon for that communication.” *Id.*

Petitioner admits that unlike *Fages*, which operates the server itself and thus has full control of the system, *Korycki* “teaches implementing its functionality in a ‘client-side’ setup.” Pet. at 40. “For that reason, *Korycki* requires the use of a daemon process to implement the desired functionality. *Fages* does not.” EX2008 at ¶146. This is also a reason why a POSITA would not be motivated to combine *Fages* with *Korycki*. *Fages* has no need for *Korycki*’s daemon based communication model.

Further, as Dr. Medvidovic explains, “none of the cited portions of *Korycki* disclose a daemon used to track active browser instances or determine whether a requested URL is already being accessed by another instance as claimed.” *Id.* at

¶147. “*Korycki*’s daemon 218 checks whether a partner web application is on an access-control list (ACL) for the purposes of communication authorization.” *Id.* (citing *Korycki* at 3:9–13 and 8:41–9:8). This disclosure does not relate to browser-instance tracking, reuse of active browser instances, or avoiding redundant browser launches. “Petitioner’s mapping equates ‘communication approval’ with ‘browser-instance management,’ but the cited prior art does not support that equivalence.” *Id.* “There is also no equivalent notion of registering or subscribing in the ’721 patent, and Petitioner doesn’t even attempt to explain what entity in the ’721 patent would subscribe to what data or channel, how that would be done, or why.” *Id.*

Finally, Petitioner applies impermissible hindsight for its motivation to combine. It imports the role of the daemon from the ’721 patent into the prior art and using that role as justification for combination. Absent the knowledge of the ’721 invention, there is no reason a POSITA would combine the cited references in the manner proposed.

Notably, this claim element was also explicitly added during prosecution. EX1002 at 363–364.

Petitioner identifies nothing in the prior art that teaches or suggests this claim element as properly understood by a POSITA at the relevant time.

e) Claim 1.d.ii: receiving the request via a browser extension associated with the first browser instance,

Petitioner admits that “*Fages*[-Jay-Hayden] does not expressly discuss a browser extension for its browsers.” Pet. at 41. “For the same reasons as stated above that *Fages* has no need to utilize a background daemon process for monitoring, *Fages* also has no need for a browser extension to facilitate communication with the daemon.” EX2008 at ¶149 (citing ’721 patent at 10:9–12). “The browser extension has two tasks, communicating with the daemon and opening new tabs requested by the daemon.” *Id.* But, “since *Fages*’ server 20 is capable of opening tabs (starting new browser instances) as needed, there is no need to implement either a daemon or a browser extension. *Id.*; see also claim element 1.d.i above.

Moreover, Dr. Medvidovic explains that “*Korycki*’s browser extension (pub-sub.js 206) serves a different technical function from the browser extension claimed in the ’721 patent.” EX2008 at ¶150. In *Korycki*, “pub-sub.js 206 implements publish-subscribe signaling for cross-browser and cross-domain messaging between web applications.” *Id.* (citing *Korycki* at 6:40–65 and 7:21–49). Unlike *Korycki*, the browser extension in the ’721 patent is used for security compartmentalization of browser instances and routing of URL-access requests based on content-class rules. *Id.* “These technical purposes are fundamentally different, and Petitioner provides no explanation for why a POSITA would replace a pub-sub messaging

component with the claim-specific security-isolation browser-extension logic of the '721 patent.” *Id.*

Further, Dr. Medvidovic explains that “adopting *Korycki*’s browser extension into the *Fages-Jai-Hayden* system would not satisfy the claimed limitation because pub-sub.js is used for messaging between applications and domains, not for class-based isolation of browser instances or class-based routing of URL requests.” *Id.* at ¶151. “A POSITA reading *Korycki* would understand pub-sub.js as a mechanism for broadcasting messages to subscribers, not for determining whether a URL belongs to a content class and steering the request to a corresponding browser instance.” *Id.* The mismatch in purpose and behavior confirms the lack of a teaching of the claimed feature.

Finally, there is no motivation to combine. *Fages, Jai, Hayden*, or *Korycki* do not identify a problem that would benefit from replacing or modifying browser-instance security partitioning with a publish-subscribe browser-messaging component. *Id.* at ¶152. Petitioner’s combination relies on hindsight: importing the role of the browser extension directly from the '721 patent and then treating it as if it were taught by the cited art.” *Id.* Absent knowledge of the '721 patent’s claimed invention, there is no reason a POSITA would implement *Korycki*’s pub-sub extension in the *Fages-Jai-Hayden* system.

f) Claim 1.d.iii: forwarding the request from the browser extension to the daemon using a native application,

Petitioner admits that “*Fages-Jai- Hayden* does not expressly discuss utilizing a ‘native application’ in its system to facilitate communication between a browser extension and daemon.” Pet. at 44–45. However, for the reasons stated regarding claim elements 1.d.i and ii above, “*Fages* has no need for a daemon or browser extension.” EX2008 at ¶153. The purpose of the native application is set forth in the ’721 patent as “forwarding messages between the daemon and the extension.” ’721 patent at 10:18–20. However, “since *Fages*’ server 20 is capable of opening tabs (starting new browser instances) as needed, there is no need to implement either a daemon or a browser extension, nor a “native application” to facilitate communications between the two.” *Id.*; *see also* claim elements 1.d.i and ii above.

Moreover, Dr. Medvidovic explains that *Korycki*’s “pub-sub.js 206 is a publish-subscribe JavaScript API implemented ‘using web-based technology (e.g., HTML5).” EX2008 at ¶154 (citing *Korycki* at 6:40–55, 7:21–33). “The ‘native’ plugins 208 and 216 are taught only as part of a publish-subscribe messaging bridge for widgets and service-provider applications but not for tracking or managing browser instances, as required by the ’721 patent.” *Id.* The cited portions of the prior art do not suggest that the pub-sub plugin 208 performs any browser-management or browser- instance tracking functionality. *Id.* (citing *Korycki* at 7:44–

46 (“provide bridges between the web applications ... and the publish-subscribe daemon application 218”)).

Petitioner further asserts that *Korycki*’s native plugin 208 performs the “same functionality” as the claimed native application. Pet. at 45 (citing *Korycki*, 6:40–65, 7:34–48). But the cited text does not support this assertion. To the contrary, it confirms that “plugin 208 merely relays publish-subscribe signaling used by widgets 204/214.” *Id.* “*Korycki* does not disclose any logic for identifying, monitoring, or correlating active browser instances, nor any structure aligned with the claim’s purpose.” *Id.* Instead, the daemon 218 in *Korycki* “implements publish-subscribe functionality” for messaging. *Korycki* at 8:44–9:8. Per Dr. Medvidovic, “this is fundamentally different from the ’721 patent’s requirement that the native application facilitate communications for the purpose of tracking browser instance activity.” *Id.*

Petitioner argues that a POSITA would have been motivated to modify *Fages-Jai-Hayden* to include *Korycki*’s native plugin 208 as a communication component. Pet. at 47 (citing *Dec.*, ¶¶277–283). But the cited portions of *Fages-Jai-Hayden* do not involve publish-subscribe signaling at all, and as Petitioner concedes, *Fages-Jai-Hayden* “does not expressly discuss” any native application. *Id.* at 44. The mere existence of a generic communication bridge in *Korycki* provides no articulated rationale for incorporating a publish-subscribe plugin into a system whose purpose

is URL classification and browser-instance comparison, not widget-to-daemon signaling.

Finally, Petitioner's motivation to combine argument assumes that any "native application" is interchangeable. As explained here, this is not correct. Petitioner makes general statements such as "native applications were well-known" or "browser-agnostic," but fails to explain how a POSITA would adapt a publish-subscribe messaging plugin to meet the '721 patent's requirement of facilitating communications tied to browser-instance tracking.

g) Claim 1.d.iv: confirming, via the daemon that the web content is not associated with the first web content class and that another browser instance is not already launched and available to access the web content; and

"For the reasons stated regarding claim element 1.d and its sub-elements, Petitioner's *Fages-Jay-Hayden* combination does not disclose use of or have a need of a daemon because it is a server side architecture." EX2008 at ¶156. "For that reason, a POSITA would not be motivated to further add *Korycki's* client side solution (daemon) to this combination." *Id.* Petitioner's combination also does not disclose "confirming. . . that another browser instance is not already launched and available to access the web content" in the relevant web content class. *Id.* "The prior art does not group web content into COI classes that may use the same browser instance." *Id.*

h) Claim 1.e: launch a new browser for accessing the web content associated with the second web content class to provide access to the URL of the second plurality of URLs via the new browser.

Although the asserted combination can launch a new browser instance, Dr. Medvidovic explains that “its purpose is not ‘for accessing the web content associated with the second web content class,’ but simply to access web content on a specific site.” EX2008 at ¶157. “The prior art does not group web content into COI classes that may use the same browser instance, and it does not teach launching a browser instance for all web content in a COI class.” *Id.*

2. Claim 2

Claim 2. The system of claim 1, wherein the new browser instance is launched automatically for display, and the web content is associated with a web server.

Claim 2 depends on claim 1 and is not disclosed or suggested for the reasons set forth regarding claim 1.

3. Motivation To Combine

A POSITA would not be motivated to combine *Fages*, *Korycki*, *Jai*, and *Hayden*.

As Dr. Medvidovic explains:

Fages discloses a server (20) capable of instantiating multiple instances of a web browser, and coordinating use thereof by the user’s remote stations. *Fages* at [0063]. The server “is capable of simultaneously executing several Internet browsers in distinct execution environments.” *Id.* “The server 20 also has a server module 40 of a

software program for the remote control of the Internet browsers executed by the server 20.” *Id.* at [0064]. “The server 20 is connected to a memory 44 containing the instructions and information needed to execute the method of FIG. 6.” *Id.* at [0065]. In particular, “the memory 44 has a buffer memory 46 and files 48 for configuring different possible Internet sessions.” *Id.* Thus, the server itself is capable of executing and tracking several Internet browsers, and their specific use.

EX2008 at ¶160. Since *Fages*’ user stations are able to directly message the server, “*Fages* has no need for a system daemon to keep track of the multiple browser instances.” *Id.* Similarly, “*Fages*’ server is capable of communicating directly with any of the browser instances.” *Id.* “This also obviates the need for a ‘browser extension’ and ‘native application’ to facilitate communications between the browser instance and the system daemon as claimed in the ’721 patent.” *Id.* “Therefore, a POSITA would not be motivated to combine *Fages* with *Korycki*.” *Id.*

Petitioner’s combination is only achieved through hindsight. Petitioner asserts that a POSITA would combine (i) *Fages*’ secure partitioning, (ii) *Jai*’s URL duplication check, and (iii) *Hayden*’s instance-reuse logic to arrive at the claimed class-based browser-instance confirmation. But Petitioner identifies no problem in the cited references which provides motivation to modify *Hayden*’s instance-selection logic to select browser instances based on web content class, nor to modify *Fages* to incorporate class-driven browser management in the first place. Petitioner’s rationale improperly assumes the very class-based browser-instance

architecture claimed in the '721 patent and uses it as the integrating glue for the combination.

B. GROUND 2: Obviousness in Light of *Fages*, *Korycki*, *Jai*, *Hayden* and *Russello*

Petitioner asserts that claims 3–5 are obvious in light of *Fages*, *Korycki*, *Jai*, *Hayden* and *Russello*.

1. Claim 3

Claim 3. The system of claim 2, wherein the processor is operable to execute an API call configured to intercept a system call to the web server, the system call to the web server intended to provide access of the URL to the first browser instance.

Claim 3 depends on claim 2 and is not disclosed or suggested for the reasons set forth regarding claim 2.

Petitioner argues that *Russello* teaches the claimed API functionality by disclosing a ptrace-based process monitor that intercepts system calls and enforces security policies, and therefore a POSITA would have been motivated to modify the *Fages-Korycki-Jai-Hayden* system to incorporate *Russello*'s ptrace API for intercepting system calls to the web server. Pet at. 53–56. Petitioner further asserts that *Russello*'s ptrace is equivalent to the '721 patent's tracer/API mechanism and would have predictably improved system security. *Id.* at 54.

As Dr. Medvidovic explains, “although *Russello* discloses intercepting system calls, Petitioner's combination of references in *Fages*' architecture does not have a

need for this functionality.” EX2008 at ¶164. As explained regarding claim element 1.c, “*Fages*’ does not disclose intercepting ‘a request to access web content’ as claimed.” *Id.* *Fages* offers a secure specialized server implementation for offering browsing security, and therefor has no need to transparently intercept system calls. *Id.* “It is not aimed at transparent compartmentalization practices on a user’s machine.” *Id.* (emphasis in original). The ’721 patent, on the other hand, offers compartmentalization which “is transparent to the user and does not require any modification of how non-sensitive content is accessed.” ’721 patent abstract. It is for this reason that the ’721 patent requires that system calls are “intercepted.” This “allows the user software to operate as normal, without modification.” *Id.* “*Fages* has no such requirement because it has full control of the entire system, server and user side.” *Id.*

Petitioner’s proposed combination lacks a specific, non-hindsight implementation rationale. As explained by Dr. Medvidovic, *Petitioner’s* combination does not explain how to

- (1) integrate OS-level master/target ptrace monitors into *Fages*’ admin-defined secure/unsecure sessions (lists L₁₁/L₂₁/L₃₁) that do not address network syscall interception at all;
- (2) adapt *Korycki*’s pub-sub daemon/plugins (daemon 218, pub-sub.js 206, plugin 208) that perform ACL/messaging—not syscall interception—into the ’721 patent’s enforcer/enabler that manages intercepted network/DNS calls; and

(3) supply the missing DNS rewrite/IP whitelist mechanisms (used by the '721 patent to act on the intercepted calls) that *Russello* does not teach. Pet at. 55–56.

EX2008 at ¶165.

The file history also confirms that Applicants emphasized ptrace interception, DNS rewrite, and the daemon/extension/native architecture, as explained in the notice of allowance. *See, e.g.*, EX1002 at 363–371. Petitioner’s generic “ptrace is known” statements do not provide the non-hindsight rationale to retrofit *Russello* into the Petitioner’s session/daemon/pub-sub system and thereby yield the '721 patent’s browser/DNS/class-driven enforcement.

2. Claim 4

Claim 4. The system of claim 3, further comprising:

- a.** wherein the processor is operable to pause operations associated with the first browser instance upon interception of the system call; and
- b.** wherein the API controls system calls made by other programs and other browser instances.

Claim 4 depends on claim 3 and is not disclosed or suggested for the reasons set forth regarding claim 3.

3. Claim 5

Claim 5. The system of claim 1, wherein the processor is operable to implement a tracer to determine whether or not a requested network object is allowed to be accessed by the first browser instance.

Claim 5 depends on claim 1 and is not disclosed or suggested for the reasons set forth regarding claim 1.

4. Motivation To Combine

A POSITA would not be motivated to combine *Fages*, *Korycki*, *Jai*, *Hayden* and *Russello*.

As explained in the “motivation to combine” section regarding ground 1, “*Fages* discloses a server (20) capable of instantiating multiple instances of a web browser, and coordinating use thereof by the user’s remote stations.” EX2008 at ¶170 (citing *Fages* at [0063]). “The server 20 also has a server module 40 of a software program for the remote control of the Internet browsers executed by the server 20.” *Fages* at [0064]. “The server 20 is connected to a memory 44 containing the instructions and information needed to execute the method of FIG. 6.” *Id.* at [0065]. Per Dr. Medvidovic, “the server itself is capable of executing and tracking several Internet browsers, and their specific use.” *Id.* *Fages* has no need to intercept system calls as taught in *Russello*, because its architecture allows for direct messages between a user terminal and server. *Id.* Similarly, “*Fages*’ server is capable of communicating directly with any of the browser instances” which “also obviates the need for intercepting system calls.” *Id.* Therefore, a POSITA would not be motivated to combine the prior art as claimed by Petitioner.

Petitioner’s combination is only achieved through hindsight. A POSITA would not be motivated to add *Russello* to Petitioner’s *Fages*-based combination

because there is simply no need to add that extra level of complexity. The functionality of *Russello*'s interceptor mechanism is already present in *Fages*.

C. GROUND 3: Obviousness in Light of *Fages*, *Korycki*, *Jai*, *Hayden*, *O'Niell*, and *Cardona*

Petitioner asserts that claims 6–8, 12–13, and 16–18 are obvious in light of *Fages*, *Korycki*, *Jai*, *Hayden*, *O'Niell*, and *Cardona*.

1. Claim 6

Claim 6. The system of claim 1, wherein the daemon comprises a multi-threaded C program that uses Unix domain sockets to communicate with the browser extension and other extensions.

Claim 6 depends on claim 1 and is not disclosed or suggested for the reasons set forth regarding claim 1.

Petitioner acknowledges that the *Fages-Korycki-Jai-Hayden* combination does not disclose this limitation. Pet. at 62. Petitioner therefore relies on *O'Niell* for “a C program daemon using Unix domain sockets” and on *Cardona* for “a multithreaded daemon.” *Id.* at 62. Petitioner thus splits the attributes of this claim across six different references, including *Fages* for the framework, *Korycki* for the “daemon” and extension/plugin context, *O'Niell* for “C” and “Unix domain sockets,” and *Cardona* for “multithreadedness.” However, pointing out various known programming language tools does not disclose the invention claimed in the '721 patent. EX2008 at ¶173.

Further, Dr. Medvidovic explains that “Unix domain sockets are local to a host, and Petitioner does not reconcile the client/server division.” *Id.* at ¶184. This is confirmed by Petitioner’s expert, who opined that Unix domain sockets “communicate only between processes on a single host” and are not network-routable. EX1003 at ¶374 (citing EX1040). The architecture employed by *Fages* is not easily compatible with that of *Korycki*. “*Fages*’ design runs the browsers on a server and allows the user stations to remotely control the server-side browsers (e.g., via NX/RDP/ICA).” EX2008 at ¶174 (citing *Fages* at [0064] and [0107]–[0108]). “*Korycki*, by contrast, places the extension (widget) and the native plugin/daemon as “native to the device” at the client device 102, with the plugin-to-daemon path over TCP.” *Id.* (citing *Korycki*, Fig. 2, 6:40–7:6, 7:21–33, 9:60–10:22).

Petitioner generically argues that these prior art systems are compatible, but does not explain where its claimed “browser extension” resides, nor how Unix domain sockets would connect the extension to the daemon. “If, consistent with *Korycki*, the extension remains on the client, Unix domain sockets cannot be used to communicate between a client-resident extension and a server-resident daemon; if the extension is moved server-side to permit Unix-domain-socket communications, Petitioner identifies no teaching or rationale for implementing *Korycki*’s pub-sub JavaScript widget and native plugin in *Fages*’ server-executed browser model or for

replacing *Korycki*'s TCP plugin↔daemon channel with Unix domain sockets in that setting. *Id.* at ¶175 (citing Pet. at 64–66).

Finally, Petitioner identifies no motivation or expectation of success of combining “*Korycki*'s extension/plugin/daemon TCP path into Unix domain sockets in *Fages*' remote-browsing architecture.” *Id.* at ¶176. Petitioner's motivation to combine is again generic—daemons are “well-known,” C is “widely utilized,” Unix domain sockets “bypass network-based sockets,” and multithreading is “beneficial.” But none of that addresses any benefit gained by the actual re-engineering required by claim 6: “replacing *Korycki*'s TCP-based plugin↔daemon transport with Unix domain sockets; co-locating the extension and daemon on the same host to make Unix domain sockets even possible; and reconciling those changes with *Fages*' server-executed browsers.” *Id.* (citing Pet. at 64–66). Petitioner's conclusory compatibility assertions do not bridge these claim-specific gaps or supply a tailored, articulated rationale with a reasonable expectation of success, neither does Dr. Zadok's declaration.

2. Claim 7

a) Claim 7(pre). A method for facilitating different categories of content to be displayed in different independent browser instances, the method comprising:

Petitioner references its argument regarding claim 1 for this claim. Pet. at 67.

Patent Owner therefore incorporates its responses thereto. This claim element is not disclosed for the reasons set forth there.

b) Claim 7.a.i: accessing data defining rules for delegating web access associated with a first content class to a first browser type and delegating web access associated with a second content class to a second browser type;

Patent owner incorporates its analysis claim element 1(b). For the reasons stated there, *Fages* does not disclose this claim element. See claim element 1(b) (discussing browser type). Petitioner asserts that a “POSITA would have understood the data defining which browser instances access sensitive versus non-sensitive web content are “rules[,]” as claimed.” Pet. at 68. This is conclusory, and Petitioner does not identify where *Fages* accesses “data defining rules” to delegate content classes to different browser types. *Id.* (citing *Dec.*, ¶389). As detailed in the claim construction section “browser instance type,” this claim limitation refers to the type of browser (e.g., Mozilla or Firefox), not the content displayed by a given browser (e.g., sensitive or general).

This claim element recites “accessing data defining rules for delegating web access associated with a first content class to a first browser type and delegating web

access associated with a second content class to a second browser type.” ’721 patent, claim 7. As Dr. Medvidovic explains, “this language requires rules that map content classes to different browser types that are accessed, not merely a general list of URLs or a general ability to launch a separate session.” EX2008 at ¶180. “The ’721 patent specification exemplifies this with an experienced-user ‘classes file’ that contains per-class ‘type descriptor’ rules (e.g., ‘F’ for Firefox, ‘C’ for Chrome) expressly delegating which browser type handles each class; these are the kind of ‘data defining rules’ that perform the claimed delegation by class to ‘browser type.’” *Id.* (citing ’721 patent at 5:51–63).

As discussed above, in *Fages*, an administrator defines secure lists L_{i1} and unsecure session S_4 , and the server blocks or launches a new browser in a different execution environment for sensitive URLs. *See, e.g., Fages* at [0085], [0090]–[0092], [0104]. This is list-based authorization and session partitioning on a server. A POSITA would not equate that with “rules” that delegate access by content class to different “browser types.” EX2008 at ¶180. In *Fages*, if a URL is on a sensitive list, the server “automatically launches the execution of another Internet browser in another execution environment,” i.e., a different session on the same browser—not application of rules delegating by class to a defined “browser type.”

The ’721 patent specification further describes the enabler/daemon workflow that accesses those rules to decide which instance (and which browser type) handles

a requested class. '721 patent at 10:12-14. This rules-based delegation mechanism is not present in *Fages*' list-and-session scheme.

c) Claim 7.a.ii: identifying a request to access an IP address by a first browser instance, the first browser instance being of the first browser type;

Petitioner references its analysis of claim 1(b) for this claim element. Pet. at 68. Patent Owner therefore incorporates its analysis of that claim element here. For the reasons stated there, *Fages* does not disclose this claim element.

Claim 7(a)(ii) requires identifying a request to access an IP address by a first browser instance of the first browser type: “identifying a request to access an IP address by a first browser instance, the first browser instance being of the first browser type.” '721 patent, claim 7. “That step focuses on IP-level access (an IP address), not merely a URL or a hostname.” EX2008 at ¶183.

The '721 patent treats IP-layer handling distinctly (e.g., DNS resolution and IP rewriting are separately claimed in claim 11), underscoring that “identifying a request to access an IP address” is not the same as noticing a URL request.

Petitioner's expert, Dr. Zadok, asserts that because *Fages* deals with URLs, a POSITA “would have understood” that identifying a URL request also identifies “a request to access an IP address,” pointing to a general description of DNS resolution. This is an inference, not a disclosure, and Petitioner cites to no portion of *Fages* that actually identifies a request to access an IP address by a browser instance.

“By contrast, *Fages*’ disclosures are URL-centric: it describes comparing a “new URL” against pre-recorded URL lists (L_{i1}/L_{i2}) and then either authorizing the connection or not authorizing it, and in a variant ‘automatically launch[ing] the execution of another Internet browser in another execution environment’ for that URL, i.e., partitioning by session on the server rather than delegating by ‘browser type.’” EX2008 at ¶184 (citing *Fages* at [0091], [0093], [0100], and [0109]). “*Fages* maps requests and enforcement at the URL level; it does not identify ‘a request to access an IP address’ at the time of the user’s request.” *Id.* *Fages* only confirms whether a URL is in a predefined list and partitions browser sessions accordingly. *See id.* (comparing S_1 , S_2 , and S_3 with S_4)). It does not use IP-address identification. *Id.*

d) Claim 7.a.iii: determining that the IP address is associated with the second content class such that the IP address should not be accessed by the first browser instance and confirming that a separate browser instance is not already in use and available to access the request, by;

Petitioner references its argument regarding claims 1(c)-1(d) and 7(a)(ii) for this claim element. Pet. at 68. Patent Owner therefore incorporates its responses thereto. This claim element is not disclosed for the reasons set forth there, as well as the reasons stated regarding the distinction between URL based and IP based web access above.

e) Claim 7.a.iii.1: implementing a daemon, the daemon comprising a multi-threaded program executed outside the first browser instance that tracks active browser instances and launches a new browser instance in order to manage according to the rules,

Petitioner references its argument regarding claims 1(d)(i) and 6 (multi-threaded program) for this claim element. Pet. at 68. Patent Owner therefore incorporates its responses thereto. This claim element is not disclosed for the reasons set forth there.

f) Claim 7.a.iii.2: receiving the request via a browser extension associated with the first browser instance,

Petitioner references its argument regarding claim 1(d)(ii) for this claim element. Pet. at 69. Patent Owner therefore incorporates its responses thereto. This claim element is not disclosed for the reasons set forth there.

g) Claim 7.a.iii.3: forwarding the request from the browser extension to the daemon using domain sockets; and

Petitioner references its argument regarding claims 1(d)(ii), 1(d)(iii) and 6 (domain sockets) for this claim element. Pet. at 69. Patent Owner therefore incorporates its responses thereto. This claim element is not disclosed for the reasons set forth there.

h) Claim 7.a.iv: generating a second browser instance to access the IP address, the second browser instance being of the second browser type.

Petitioner references its argument regarding claims 1(e) and 7(a)(ii) for this claim element. Pet. at 69. Patent Owner therefore incorporates its responses thereto. This claim element is not disclosed for the reasons set forth there.

3. Claim 8

Claim 8. The method of claim 7, wherein the second browser instance is launched automatically to a client device for display.

Claim 8 depends on claim 7 and is not disclosed or suggested for the reasons set forth regarding claim 7.

Petitioner references its argument regarding claims 1 and 7 for this claim. Pet. at 69. Patent Owner therefore incorporates its responses thereto. Claim 8 is not disclosed for the reasons set forth there.

4. Claim 12

a) Claim 12(pre). A non-transitory computer readable medium encoded with a set of instructions for facilitating different categories of content to be displayed in different independent browser instances, the set of instructions, executable by a processor, comprising:

Petitioner references its argument regarding claims 1(pre) and 7(pre) and (a) for this claim element. Pet. at 69. Patent Owner therefore incorporates its responses thereto. This claim element is not disclosed for the reasons set forth there.

b) Claim 12.a: accessing data associating a first web content class with a first browser type and a second web content class with a second browser type;

Petitioner references its argument regarding claim 7(a)(i) for this claim element. Pet. at 69. Patent Owner therefore incorporates its responses thereto. This claim element is not disclosed for the reasons set forth there.

c) Claim 12.b: accessing a request to access web content via a first browser instance, the first browser instance being of the first browser type; and

Petitioner references its argument regarding claim 7(a)(ii) for this claim element. Pet. at 69. Patent Owner therefore incorporates its responses thereto. This claim element is not disclosed for the reasons set forth there.

d) Claim 12.c: comparing the web content to the first web content class and the second web content class to determine whether the web content is accessible to the first browser instance, by

Petitioner references its argument regarding claims 1(c) and 7(a)(iii) for this claim element. Pet. at 69. Patent Owner therefore incorporates its responses thereto. This claim element is not disclosed for the reasons set forth there.

e) Claim 12.c.i: implementing a daemon, the daemon comprising a multi-threaded program executed outside the first browser instance that tracks active browser instances and launches a new browser instance in order to manage according to the rules,

Petitioner references its argument regarding claim 7(a)(i) and 7(a)(iii)(1) for this claim element. Pet. at 70. Patent Owner therefore incorporates its responses thereto. This claim element is not disclosed for the reasons set forth there.

f) Claim 12.c.ii: receiving the request via a browser extension associated with the first browser instance,

Petitioner references its argument regarding claim 7(a)(iii)(2) for this claim element. Pet. at 70. Patent Owner therefore incorporates its responses thereto. This claim element is not disclosed for the reasons set forth there.

g) Claim 12.c.iii: forwarding the request from the browser extension to the daemon using domain sockets.

Petitioner references its argument regarding claim 7(a)(iii)(3) for this claim element. Pet. at 70. Patent Owner therefore incorporates its responses thereto. This claim element is not disclosed for the reasons set forth there.

5. Claim 13

Claim 13. The non-transitory computer readable medium of claim 12, further comprising:

Claim 13 depends on claim 12 and is not disclosed or suggested for the reasons set forth regarding claim 12.

a) Claim 13(pre). The non-transitory computer readable medium of claim 12, further comprising:

Petitioner references its arguments regarding claim 12(pre). Pet. at 70. Patent Owner therefore incorporates its responses thereto. This claim element is not disclosed for the reasons set forth there.

b) Claim 13.a: generating a second browser instance to access the web content,

Petitioner references its arguments regarding claims 1(e) and 7(a)(iv) for this claim element. Pet. at 70. Patent Owner therefore incorporates its responses thereto. This claim element is not disclosed for the reasons set forth there.

c) Claim 13.b: wherein the second browser instance is launched automatically to a client device for display.

Petitioner references its arguments regarding claims 2 and 8 for this claim element. Pet. at 70. Patent Owner therefore incorporates its responses thereto. This claim element is not disclosed for the reasons set forth there.

6. Claim 16

Claim 16. The non-transitory computer readable medium of claim 12, wherein the second web content class is associated with sensitive information indicating that accessing the web content by the first browser instance presents a security risk.

Claim 16 depends on claim 12 and is not disclosed or suggested for the reasons set forth regarding claim 12.

Additionally, Petitioner references its arguments regarding claims 1(a) and 12 for this claim. Pet. at 70. Patent Owner therefore incorporates its responses thereto. Claim 16 is not disclosed for the reasons set forth there.

7. Claim 17

Claim 17. The non-transitory computer readable medium of claim 12, wherein the first web content class is associated with a first plurality of URLs and the second web content class is associated with a second plurality of URLs, the first plurality of URLs being different from the second plurality of URLs.

Claim 17 depends on claim 12 and is not disclosed or suggested for the reasons set forth regarding claim 12.

Additionally, Petitioner references its arguments regarding claims 1(a) and 12 for this claim. Pet. at 70. Patent Owner therefore incorporates its responses thereto. Claim 17 is not disclosed for the reasons set forth there.

8. Claim 18

Claim 18. The non-transitory computer readable medium of claim 12, wherein the data further defines rules limiting web access by the first browser instance to URLs defined by the first web content class.

Claim 18 depends on claim 12 and is not disclosed or suggested for the reasons set forth regarding claim 12.

Additionally, Petitioner references its arguments regarding claims 7(a)(i) and 12 for this claim. Pet. at 70. Patent Owner therefore incorporates its responses thereto. Claim 18 is not disclosed for the reasons set forth there.

9. Motivation To Combine

A POSITA would not be motivated to combine *Fages*, *Korycki*, *Jai*, *Hayden*, *O’Niell*, and *Cardona*.

As explained by Dr. Medvidovic regarding claim 6 and confirmed by Petitioner’s expert, “Unix domain sockets are local to a host and are not network-routable. EX2008 at ¶210; EX1003 at ¶374 (both citing EX1040). Petitioner does not address the issues associated with the client/server architecture. As Dr. Medvidovic explains:

Fages’ design runs the browsers on a server and allows the user stations to remotely control the server-side browsers (e.g., via NX/RDP/ICA). See, e.g., *Fages* at [0064], [0107]–[0108]. *Korycki*, by contrast, places the extension (widget) and the native plugin/daemon as “native to the device” at the client device 102, with the plugin-to-daemon path over TCP. See *Korycki* at Fig. 2, 6:40–7:6, 7:21–33, 9:60–10:22. A POSITA would not be motivated to combine these due to the architectural differences, and the technical difficulties required to overcome the same.

EX2008 at ¶210.

Petitioner’s generic assertion of “compatibility” does not explain where the “browser extension” resides in its Ground 3 configuration and how Unix domain sockets would actually connect the extension to the daemon in that combined environment. If, consistent with *Korycki*, the extension remains on the client-side, Unix domain sockets cannot be used to communicate between a client-resident extension and a server-resident daemon; if the extension is moved server-side to

permit Unix-domain-socket communications, Petitioner identifies no teaching or rationale for implementing *Korycki*'s pub-sub JavaScript widget and native plugin in *Fages*' server-executed browser model or for replacing *Korycki*'s TCP plugin↔daemon channel with Unix domain sockets in that setting. Pet. at 64–66.

Petitioner identifies no reasonable motivation to solve a known problem or expectation of success to re-plumb “*Korycki*'s extension/plugin/daemon TCP path into Unix domain sockets in *Fages*' remote-browsing architecture.” EX2008 at ¶211. Petitioner's asserted motivations to combine are again generic—daemons are “well-known,” C is “widely utilized,” Unix domain sockets “bypass network-based sockets,” and multithreading is “beneficial.” But none of that identifies a benefit a POSITA would have expected to gain by combining these references to achieve the invention of the '721 patent—until the invention of the '721 patent, that is.

D. GROUND 4: Obviousness in Light of *Fages*, *Korycki*, *Jai*, *Hayden*, *O'Niell*, *Cardona*, and *Russello*

Petitioner asserts that claims 9-10 and 14-15 are obvious in light of *Fages*, *Korycki*, *Jai*, *Hayden*, *O'Niell*, *Cardona*, and *Russello*.

1. Claim 9

Claim 9. The method of claim 7, further comprising: executing an API call configured to intercept a system call to a web server associated with the IP address, the system call initiated by the first browser instance.

Claim 9 depends on claim 7 and is not disclosed or suggested for the reasons set forth regarding claim 7.

Additionally, Petitioner references its arguments regarding claims 3 and 7(a)(ii) for this claim. Pet. at 71. Patent Owner therefore incorporates its responses thereto. Claim 9 is not disclosed for the reasons set forth there.

2. Claim 10

Claim 10. The method of claim 7, further comprising: executing a tracer to determine whether or not a requested network object is allowed to be accessed by the first browser instance.

Claim 10 depends on claim 7 and is not disclosed or suggested for the reasons set forth regarding claim 7.

Additionally, Petitioner references its arguments regarding claims 5 and 7 for this claim. Pet. at 71. Patent Owner therefore incorporates its responses thereto. Claim 10 is not disclosed for the reasons set forth there.

3. Claim 14

Claim 14 depends on claim 12 and is not disclosed or suggested for the reasons set forth regarding claim 12.

a) Claim 14(pre): The non-transitory computer readable medium of claim 12, further comprising:

Petitioner references its argument regarding claim 12(pre) for this claim element. Pet. at 71. Patent Owner therefore incorporates its responses thereto. This claim element is not disclosed for the reasons set forth there.

b) Claim 14.a: executing an API call configured to intercept a system call to a web server,

Petitioner references its arguments regarding claims 3 and 12 for this claim element. Pet. at 71. Patent Owner therefore incorporates its responses thereto. This claim element is not disclosed for the reasons set forth there.

c) Claim 14.b: the system call associated with the first browser instance.

Petitioner references its arguments regarding claims 3 and 12 for this claim element. Pet. at 71. Patent Owner therefore incorporates its responses thereto. This claim element is not disclosed for the reasons set forth there.

4. Claim 15

Claim 15. The non-transitory computer readable medium of claim 12, further comprising: executing a tracer to determine whether or not the first browser instance is authorized to access a requested network.

Claim 15 depends on claim 12 and is not disclosed or suggested for the reasons set forth regarding claim 12.

Additionally, Petitioner references its arguments regarding claims 5, 10 and 12 for this claim. Pet. at 71. Patent Owner therefore incorporates its responses thereto. Claim 15 is not disclosed for the reasons set forth there.

A. GROUND 5: Obviousness in Light of *Fages, Korycki, Jai, Hayden, O’Niell, Cardona, and Tazuma*

Petitioner asserts that claim 11 is obvious in light of *Fages, Korycki, Jai, Hayden, O’ Niell, Cardona* and *Tazuma*.

1. Claim 11

Claim 11. The method of claim 7, further comprising:

Claim 11 depends on claim 7 and is not disclosed or suggested for the reasons set forth regarding claim 7.

a) Claim 11(pre): The method of claim 7, further comprising:

Petitioner references its argument regarding claim 7(pre) for this claim element. Pet. at 72. Patent Owner therefore incorporates its responses thereto. This claim element is not disclosed for the reasons set forth there.

b) Claim 11.a: generating a domain name system (DNS) query to resolve the IP address;

Petitioner references its argument regarding claim 7(a)(ii) for this claim element. Pet. at 72. Patent Owner therefore incorporates its responses thereto. This claim element is not disclosed for the reasons set forth there.

Petitioner further relies on *Tazuma* to supply a disclosure of a computer “performing a [DNS] query to determine the IP address of the destination server.” *Id.* (quoting *Tazuma*’s description that “DNS is a standard IP protocol and service that can be used to translate domain names (e.g., www.yahoo.com) to IP addresses (e.g. 200.35.53.2, etc.).”). This disclosure does not teach or suggest this claim element for several reasons.

First, Petitioner ignores that its DNS-query generation does not start with a specific “IP address.” Claim 7(a)(ii) requires “identifying a request to access an IP address by a first browser instance,” which begins with an IP address. Petitioner’s cross-reference to claim element 7(a)(ii) does not explain how, after a request “to access an IP address” has been identified, the method then “generat[es] a DNS query to resolve the IP address.” ’721 patent, claim 11. “DNS queries conventionally translate domain names into IP addresses, and do not ‘resolve’ an already-specified IP address.” EX2008 at ¶228 (citing Pet. at 72). *Tazuma*’s DNS server converting domain names into IP addresses does not automatically disclose the inverse of that operation, i.e., to resolve an IP address to a URL as in claim element 7(a)(ii). *Id.* “*Tazuma*’s disclosure is about resolving a domain name into an IP address, not about issuing a DNS query to ‘resolve’ an IP address that is already known.” *Id.* (citing *Tazuma*, 8:38-49).

Intrinsic evidence underscores the materiality and specificity of the DNS limitations. The ’721 patent describes DNS handling in the context of controlling access by rewriting IP addresses in DNS responses to prevent access to restricted websites, confirming that its DNS logic is not a generic background step but a particular access-control mechanism integrated with the claimed class-based browser management. ’721 patent at 9:41–56. During prosecution, the Applicant added what is now claim 11 (then claim 20) to recite, *inter alia*, generating a DNS

query, issuing a DNS response with a rewritten IP address, and returning the rewritten IP address—limitations the Examiner viewed as not expressly taught by the earlier-cited art and that were addressed with separate DNS references, further confirming that these DNS steps were not assumed background but discrete limitations. EX1002 at 310, 342. Petitioner’s treatment—equating claim 11(a) to a general statement that DNS translates names to IPs—does not meet the claim’s specific requirement in the method’s context.

“Although *Tazuma* teaches that a client may ‘perform[] a DNS query to determine the IP address of the destination server’ when starting from a domain name, its system architecture repeatedly emphasizes that a gateway device’s packet filter can retarget DNS query packets to an internal nameserver and that nameserver components resolve, cache, or even reply with ‘fake’ IP addresses, depending on rules, context, and link state.” EX2008 at ¶229 (citing *Tazuma* at 8:38–40, 8:52–60, 9:40–43). *Tazuma* does not disclose the reverse of this operation—i.e. returning a URL from an IP address—as claimed here.

c) Claim 11.b: determining that the IP address is associated with the second content class; and

Petitioner references its argument regarding claim 7(a)(iii) for this claim element. Pet. at 73. Patent Owner therefore incorporates its responses thereto. This claim element is not disclosed for the reasons set forth there.

d) Claim 11.c: issuing a DNS response, the IP address being rewritten in the DNS response to prevent the first browser instance from accessing the IP address and information of the second content class; and

Petitioner references its argument regarding claim 7(a)(iii) for this claim element, and generally to its arguments regarding claims 1 and 7 for the proposition that “*Fages* teaches utilizing multiple independent browsers where different browsers and sessions are related to first and second content classes.” Pet. at 73. Patent Owner therefore incorporates its responses thereto. This claim element is not disclosed for the reasons set forth there.

IX. CONCLUSION

The Board should deny institution because CrowdStrike has failed to demonstrate a reasonable likelihood of prevailing as to any challenged claim.

Dated: December 23, 2025

Respectfully submitted,

/s/ Jay P. Kesan

Jay P. Kesan (Reg. No. 37,488)
KEY KESAN DALLMANN PLLC
1050 Connecticut Avenue, N.W. Suite
500 Washington, D.C. 20036
jay.kesan@kkd-law.com
Telephone: (202) 772-1100

Alfonso G. Chan
Andrew Cochran
Cahill Gordon & Reindel LLP
900 16th Street, N.W., Suite 500
Washington, D.C. 20006

Telephone: (202) 862-8942

Attorneys for Patent Owner
SKYSONG INNOVATIONS, LLC

CERTIFICATE OF SERVICE

Pursuant to 37 C.F.R. §§ 42.6(e)(4) and 42.25(b), the undersigned certifies that on 23 December 2025, a complete copy of Patent Owner's Preliminary Response was filed electronically through the Patent Trial and Appeal Board's PTABE2E System and provided, via electronic service, to the Petitioner by serving the correspondence address of record.

Dated: December 23, 2025

/s/ Jay P. Kesan
Jay P. Kesan

CERTIFICATE OF WORD COUNT

Pursuant to 37 C.F.R. § 42.24, the undersigned certifies that the foregoing Patent Owner's Preliminary Response contains less than 14,000 words excluding the table of contents, table of authorities, certificate of service, certificate of word count, and appendix of exhibits. Patent Owner has relied on the word count feature of the word processing software used to create this paper in making this certification.

Dated: December 23, 2025

/s/ Jay P. Kesan
Jay P. Kesan